# How Much for the Elephants?

It's a weird profession we have chosen for a living, right? I mean, after all, we work in a profession that considers a millisecond a very long time, and we consider a 128MB USB thumb drive (which, after all, holds almost 100 1.44MB floppies) totally obsolete ($9.95 on sale at a local computer store over Christmas). Ours is a profession where new computer languages come and go yearly, yet COBOL, one of the most commonly used programming languages in the world, still remains a language standardized in 1960 but with its roots embedded in the early 1950s (making it older than me!).

I have been teaching computer science for more than 30 years (first as a teaching assistant in 1974 at the University of Central Florida), starting back when there was barely a discipline known as software engineering[1]. While some things in the field of software engineering come and go, some *truths* need to be relearned by each generation.

1. No programming language ever developed will make it the least bit difficult to write a horrible program[2]!
2. You really can't complete a project until you know the requirements.
3. The first set of requirements is almost never the right requirements.
4. Neither are the second, third, or probably the fourth.
5. The final set of requirements isn't.
6. No matter how good a coder you are, you need a design.
7. Code that is so simple it can't go wrong – will.
8. There is always one error that error-checking routines will miss.
9. No matter what the problem is, it's usually management.
10. No matter how simple it is – you have to test it.
11. Everybody else writes code that needs testing. They say the same about you.
12. Almost any shortcuts you take to speed up the project make it take longer.
13. It's always going to take longer (and cost more) than you plan, even when you take, "It's always going to take longer (and cost more) than you plan" into account.

No getting around it – what we do for a living is hard. No. 13 is particularly difficult. How long does it take? How much will it cost? Even the most experienced developers are often *so* far off with their initial estimates.

Back in college, you never really knew how long a particular programming assignment was going to take. Some that appeared really easy turned out to be really hard (debugging pointers almost *always* involved more work than you thought). And, some jobs that appeared to be really hard turned out to take almost no time at all (the *quick sort* took what, about 10 lines of code?)

The roots of cost (and time) estimation go back a long way. I am reasonably sure that Hannibal, as he was planning to cross the Alps in 219 B.C. during the Second Punic War, was somehow thinking of the incremental cost of each additional elephant. It is interesting to note that the crossing of the Alps with elephants, the event that Hannibal is so famous for, was not really a success. He started out with 34 elephants, but lost many of the elephants on the crossing, and all but one were dead by the end of the Battle of Trebbia[3].

It is also interesting to note that Hannibal, while winning important battles, was beset by political jealousies at home, and this eventually proved his undoing. Because he was unable to get the necessary equipment and personnel, he was not able to take advantage of opportunities and his victories turned into a failure[4]. I could easily draw parallels between Hannibal and many other modern-day software project managers (especially the political jealousies), except for the fact that at around age 70, Hannibal committed suicide rather than face humiliation at the hands of his enemies (we offer early retirement as an option).

Hannibal is famous for the *elephant crossing*, yet the elephants proved to be of limited usefulness during the actual war. What caused Hannibal's eventual downfall was more simplistic – siege equipment (hardware) and people – are basic factors in cost estimation. Would you rather be famous, or succeed? If you want to be famous, see if you can convince 34 elephants to help you code your project in Visual C++. If you would rather succeed, why not have an accurate estimate of costs?

For one final comment on cost estimation, I would like to add that it is obviously a long-standing tradition to mock those whose projects fail due to a lack of cost estimation. In fact, such mockery of those committing cost-estimation failures is reliably documented:

> For which of you, desiring to build a tower, does not first sit down and count the cost, whether he has enough to complete it? Otherwise, when he has laid a foundation, and is not able to finish, all who see it begin to mock him, saying, "This man began to build, and was not able to finish[5]."

When your management suggests that cost estimation has been ordained from on high, you thought they just meant the Pentagon, right?

Hope to see you at SSTC 2005. It's well worth the cost!

**— David A. Cook, Ph.D.**
Senior Research Scientist
The AEgis Technologies Group, Inc.
dcook@aegistg.com

## Can You BACKTALK?

Here is your chance to make your point, even if it is a bit tongue-in-cheek, without your boss censoring your writing. In addition to accepting articles that relate to software engineering for publication in CROSSTALK, we also accept articles for the BACKTALK column. BACKTALK articles should provide a concise, clever, humorous, and insightful perspective on the software engineering profession or industry or a portion of it. Your BACKTALK article should be entertaining and clever or original in concept, design, or delivery. The length should not exceed 750 words.

For a complete author's packet detailing how to submit your BACKTALK article, visit our Web site at <www.stsc.hill.af.mil>.

1. For you purists, the NATO Science Committee sponsored two conferences on software engineering in 1968 and 1969, which many feel gave the field its initial boost. Many also believe these conferences marked the official start of the profession. The term *software engineering* has been used since the late 1950s. See <http://en.wikipedia.org/wiki/History_of_software_engineering>.
2. I make no claim as to the originality of these *truths*. No. 1, for example, comes from "There does not now, nor will there ever, exist a programming language in which it is the least bit hard to write bad programs," a quote by Lawrence Flon in "On Research in Structured Programming," SIGPLAN Notices 10:10 (Oct. 1975). This truism is proved again and again as newer and newer languages are developed.
3. See <www.barca.fsnet.co.uk/elephants.htm>.
4. See <www.carpenoctem.tv/military/hannibal.html>.
5. The Bible. Luke 14:28. Revised Standard Version.